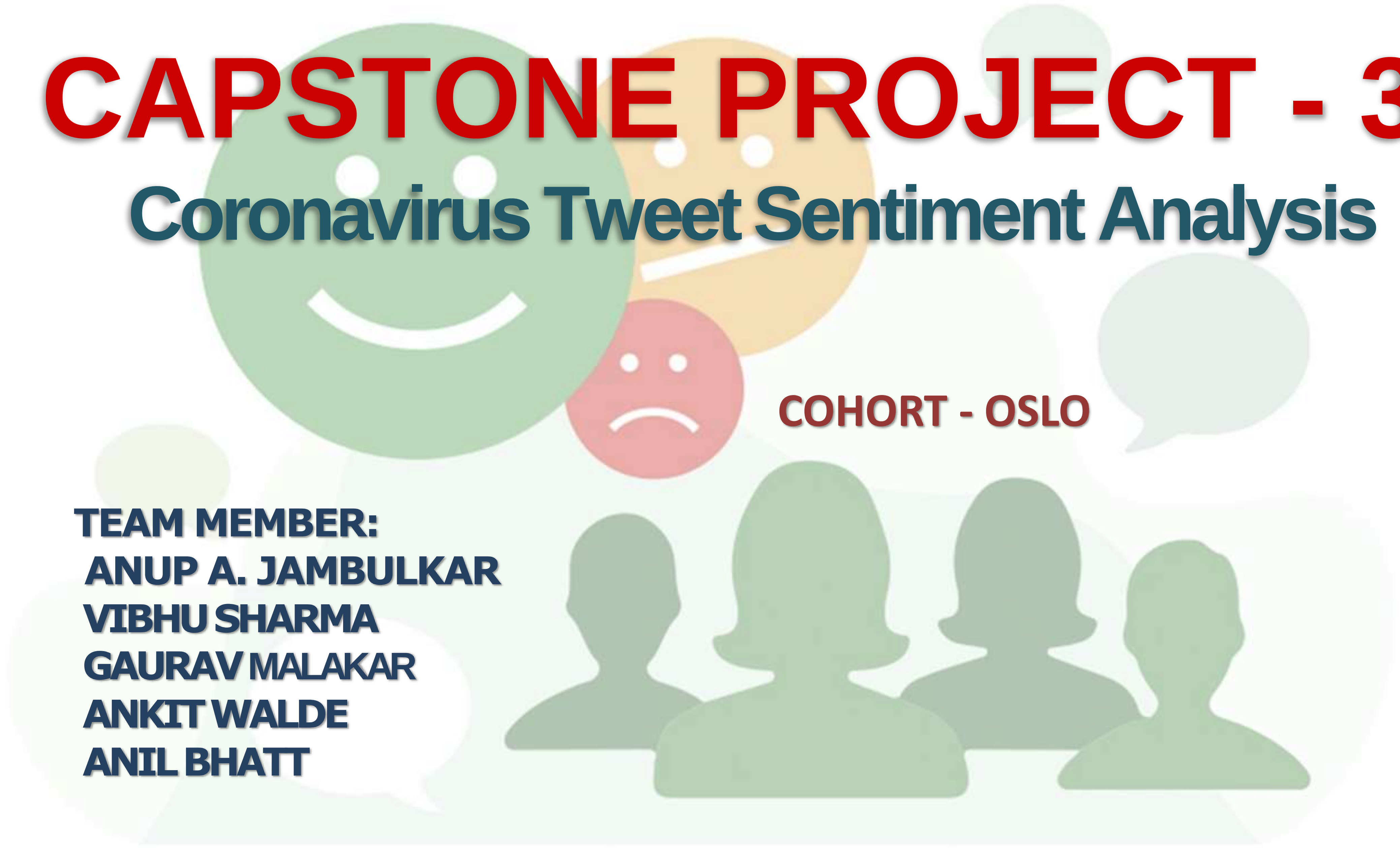


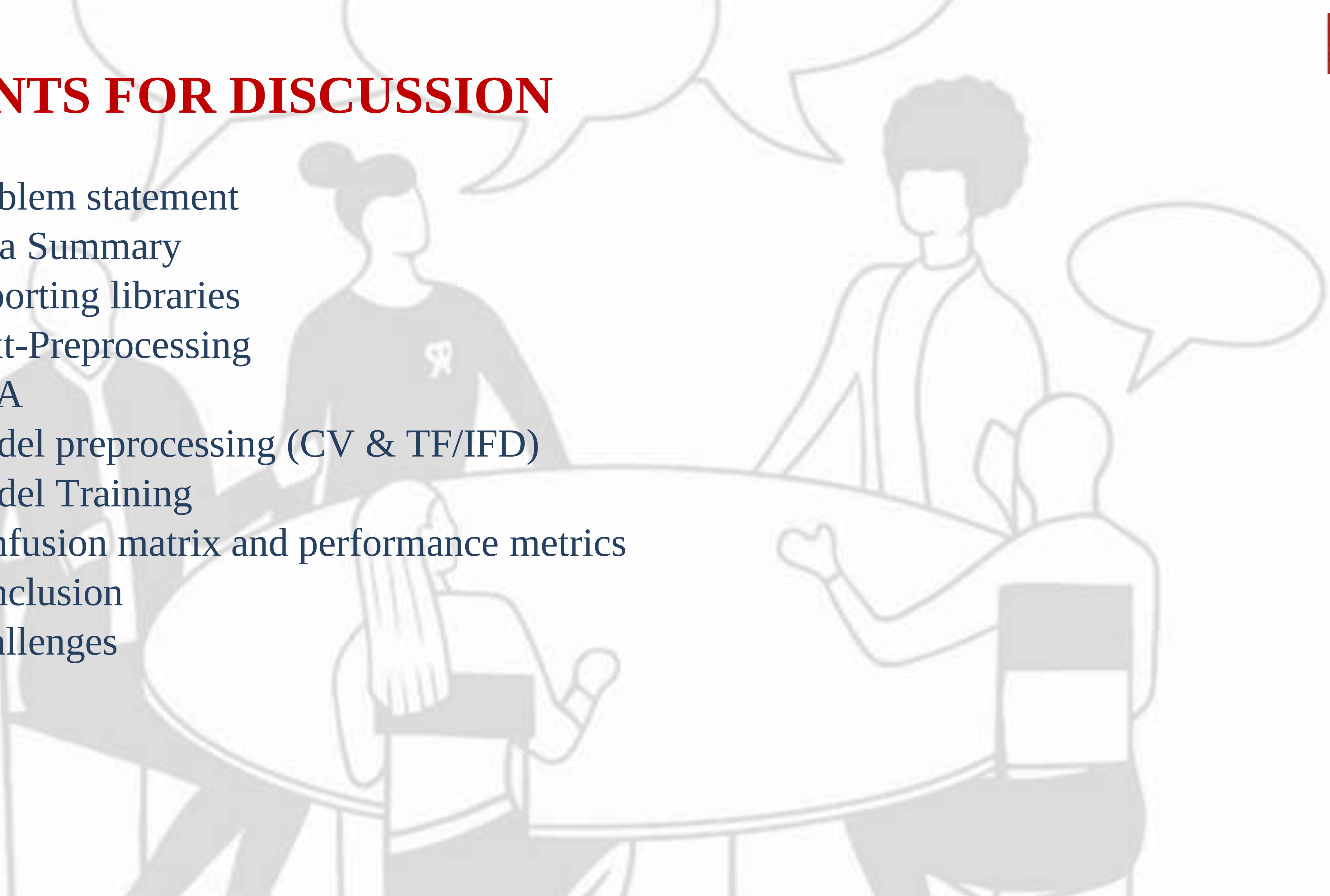
CAPSTONE PROJECT - 3

Coronavirus Tweet Sentiment Analysis

COHORT - OSLO

TEAM MEMBER:

ANUP A. JAMBULKAR
VIBHU SHARMA
GAURAV MALAKAR
ANKIT WALDE
ANIL BHATT



POINTS FOR DISCUSSION

- Problem statement
- Data Summary
- Importing libraries
- Text-Preprocessing
- EDA
- Model preprocessing (CV & TF/IFD)
- Model Training
- Confusion matrix and performance metrics
- Conclusion
- Challenges

Problem Description

This challenge asks us to build a classification model to predict the sentiment of COVID-19 tweets. The tweets have been pulled from Twitter and manual tagging has been done then.

The names and usernames have been given codes to avoid any privacy concerns.

We are given the following information:

1. Location
2. Tweet At
3. Original Tweet
4. Label

Data Summary

We are given the dataset having 6 columns –

- UserName
- Screenname
- Location
- TweetAt
- OriginalTweet
- Sentiment

Importing Libraries & Data Inspection

- Pandas – Manipulation of tabular data in Dataframes
- Numpy – Mathematical operations on arrays
- Matplotlib – Visualization
- Seaborn – Visualization
- Sklearn – Data Modeling
- Nltk – Pre Processing / Feature Engineering
- WordCloud – Visualization

Text Pre-processing

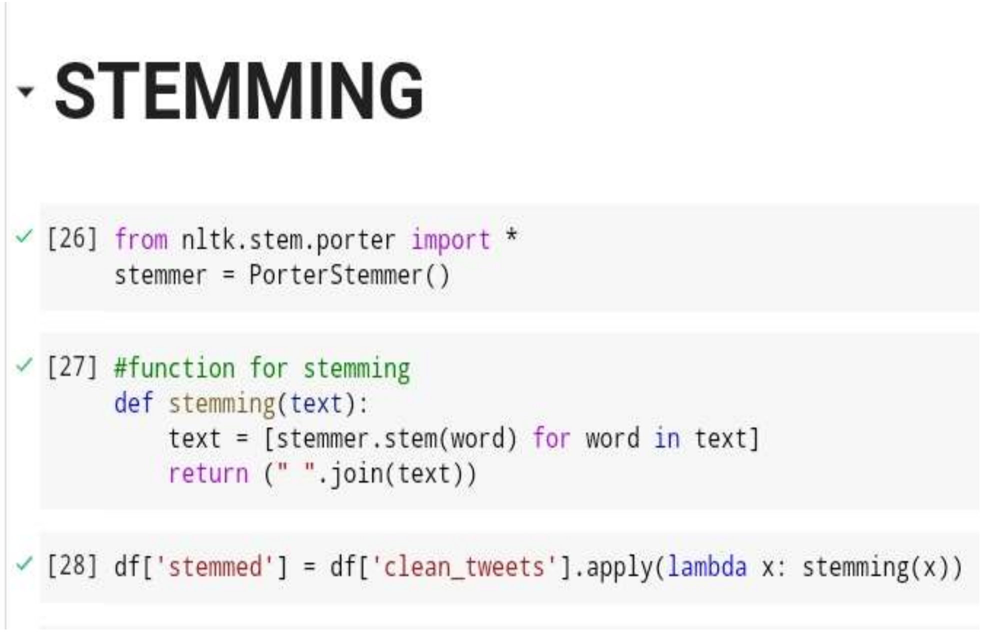
Step 1 : Converted all characters to lowercase.

Step 2 : Removed Punctuation.

Step 3 : Removed stop words.

Step 4: Stemming

Step 5: Lemmatizing

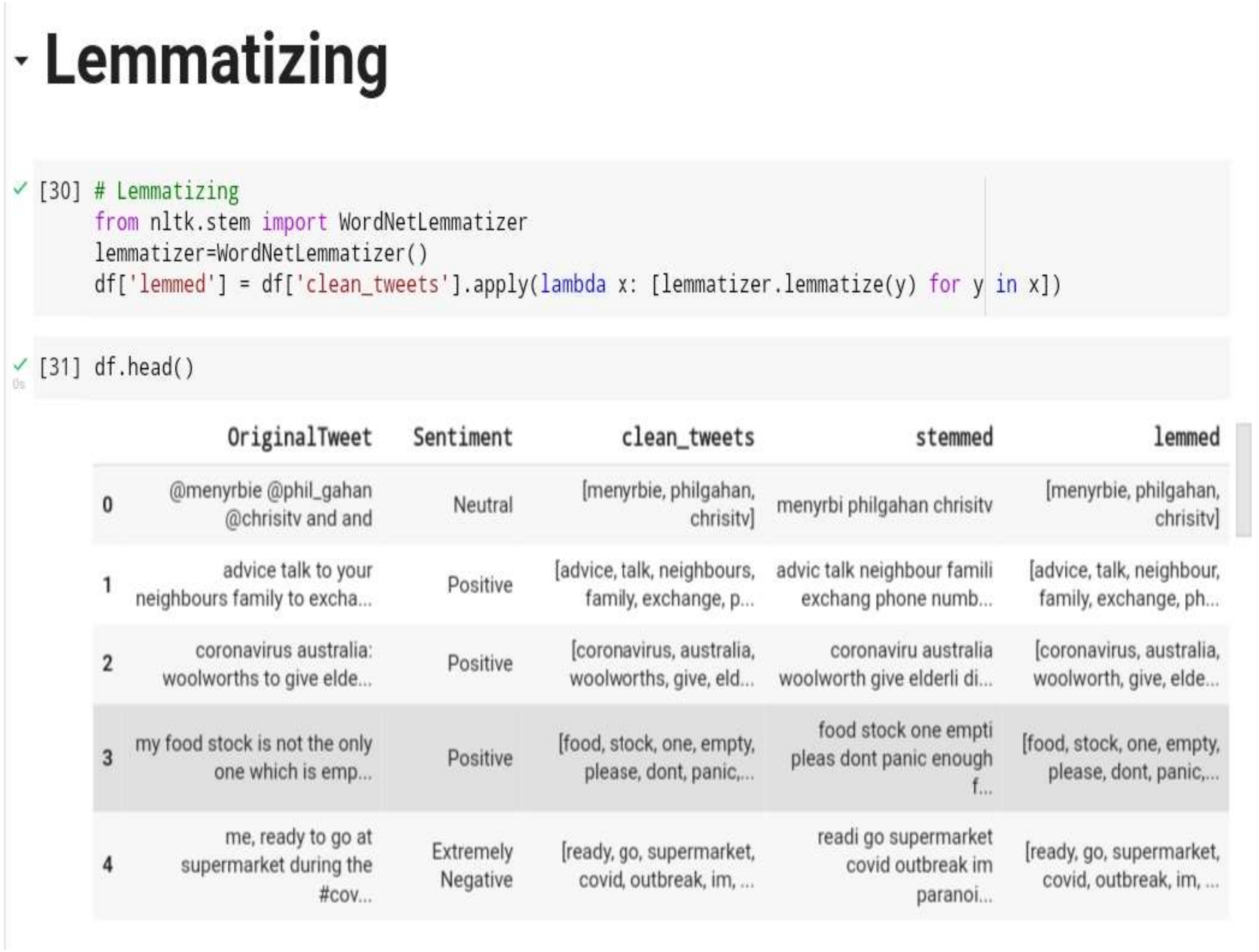


STEMMING

```
[26] from nltk.stem.porter import *
     stemmer = PorterStemmer()
```

```
[27] #function for stemming
     def stemming(text):
         text = [stemmer.stem(word) for word in text]
         return (" ".join(text))
```

```
[28] df['stemmed'] = df['clean_tweets'].apply(lambda x: stemming(x))
```



Lemmatizing

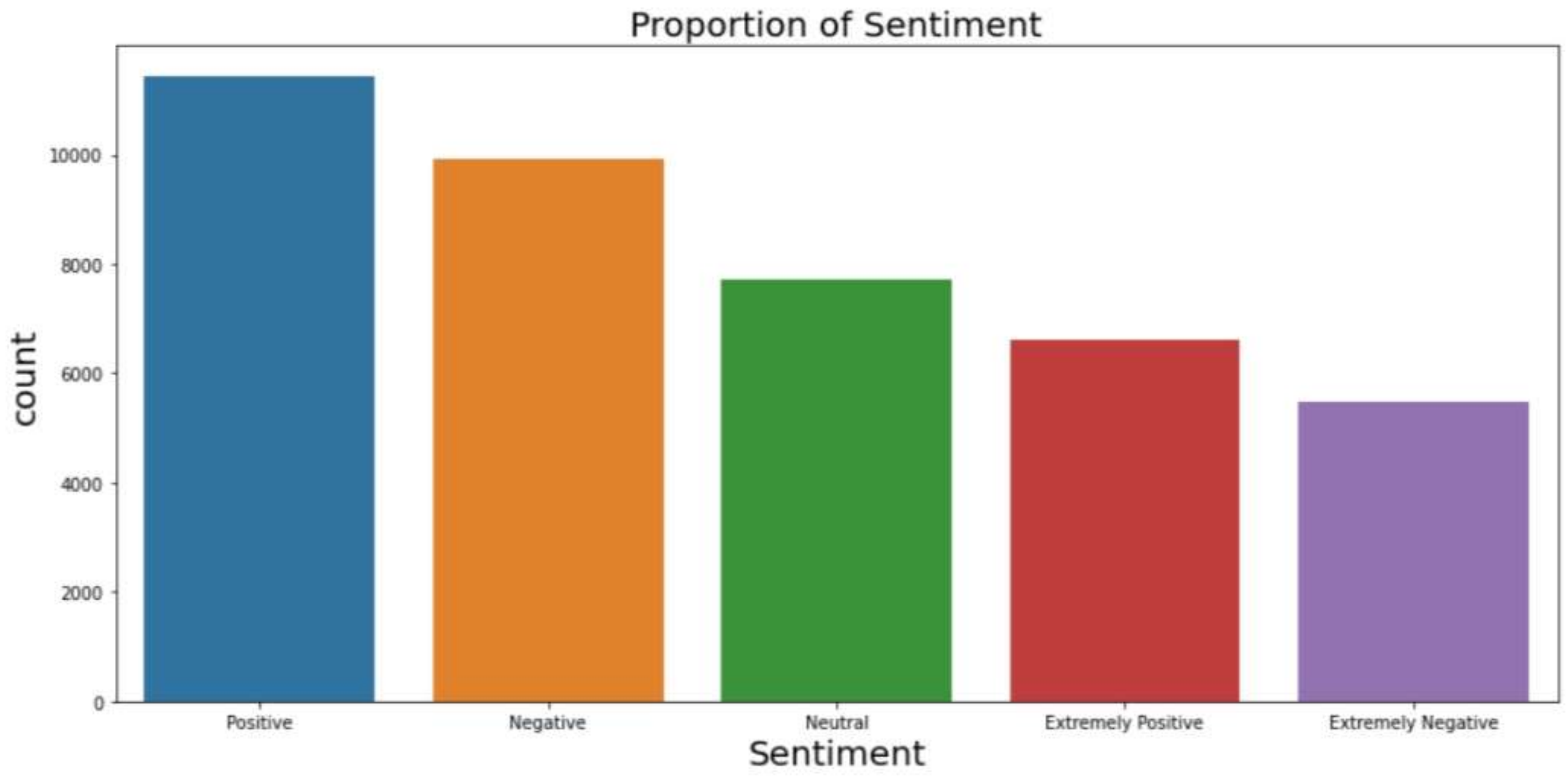
```
[30] # Lemmatizing
     from nltk.stem import WordNetLemmatizer
     lemmatizer=WordNetLemmatizer()
     df['lemmed'] = df['clean_tweets'].apply(lambda x: [lemmatizer.lemmatize(y) for y in x])
```

```
[31] df.head()
```

	OriginalTweet	Sentiment	clean_tweets	stemmed	lemmed
0	@menyrbie @phil_gahan @chrisitv and and	Neutral	[menyrbie, philgahan, chrisitv]	menyrbi philgahan chrisitv	[menyrbie, philgahan, chrisitv]
1	advice talk to your neighbours family to excha...	Positive	[advice, talk, neighbours, family, exchange, p...	advic talk neighbour famili exchang phone numb...	[advice, talk, neighbour, family, exchange, ph...
2	coronavirus australia: woolworths to give elde...	Positive	[coronavirus, australia, woolworths, give, eld...	coronaviru australia woolworth give elderli di...	[coronavirus, australia, woolworth, give, elde...
3	my food stock is not the only one which is emp...	Positive	[food, stock, one, empty, please, dont, panic,...	food stock one empti pleas dont panic enough f...	[food, stock, one, empty, please, dont, panic,...
4	me, ready to go at supermarket during the #cov...	Extremely Negative	[ready, go, supermarket, covid, outbreak, im, ...	readi go supermarket covid outbreak im paranoi...	[ready, go, supermarket, covid, outbreak, im, ...

Exploratory Data Analysis

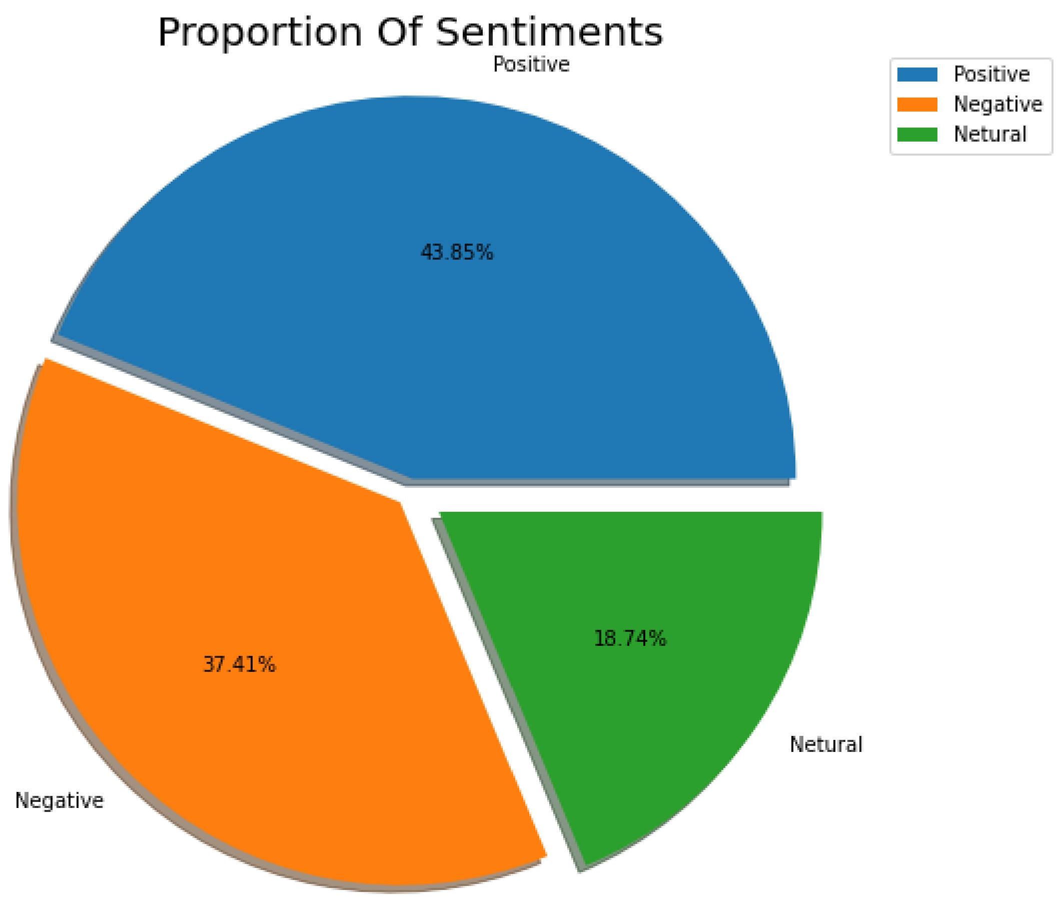
Sentiments



There are five types of sentiments. The above graph shows count of each sentiment.

EDA Continued…

New Sentiments



As there was five types of sentiments – Positive Sentiment, Extremely Positive Sentiment, Negative Sentiment, Extremely Negative Sentiment and Neutral Sentiment. So, we have replaced Extremely Positive Sentiment by Positive Sentiment and Extremely Negative Sentiment by Negative Sentiment. Now we have three types of sentiments – Positive Sentiment, Negative Sentiment and Neutral Sentiment.

The Pi Chart shows the proportion of each sentiment.

There are 43.85% Positive Sentiments,37.41% Negative Sentiments and 18.74% Neutral Sentiments.

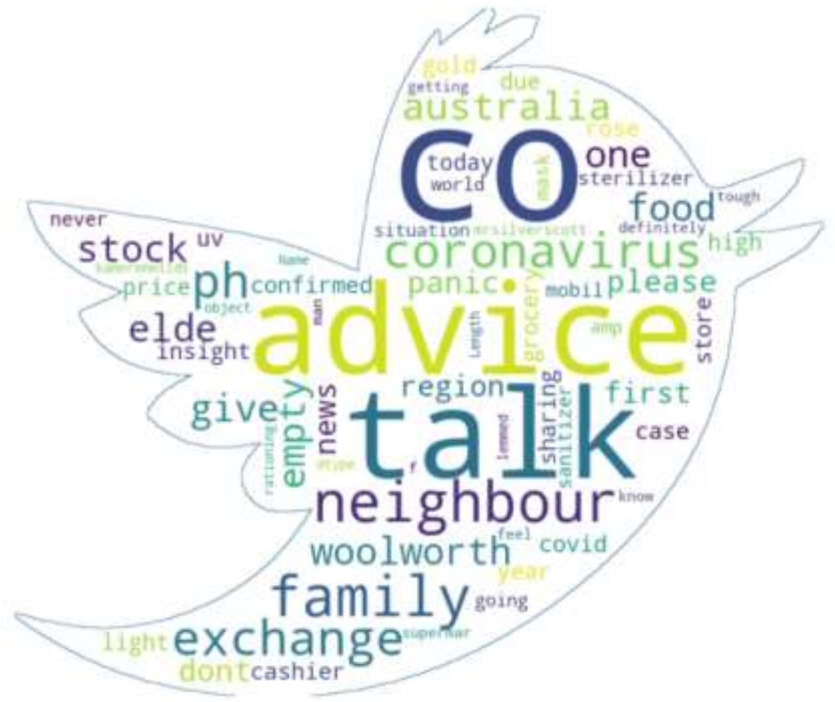
Positive Sentiments are having higher proportion among all.

EDA Continued…

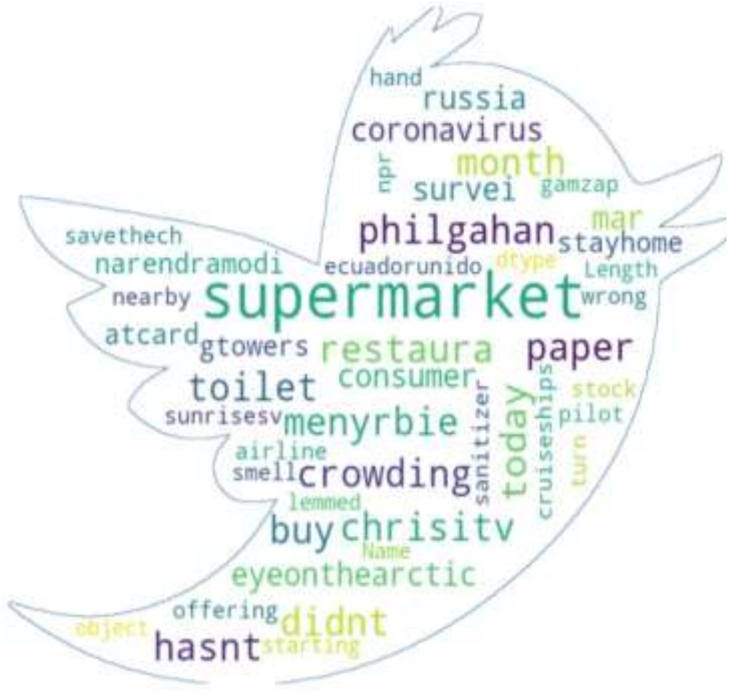
Word cloud

Word Clouds are **visual displays of text data – simple text analysis**. Word Clouds display the most prominent or frequent words in a body of text.

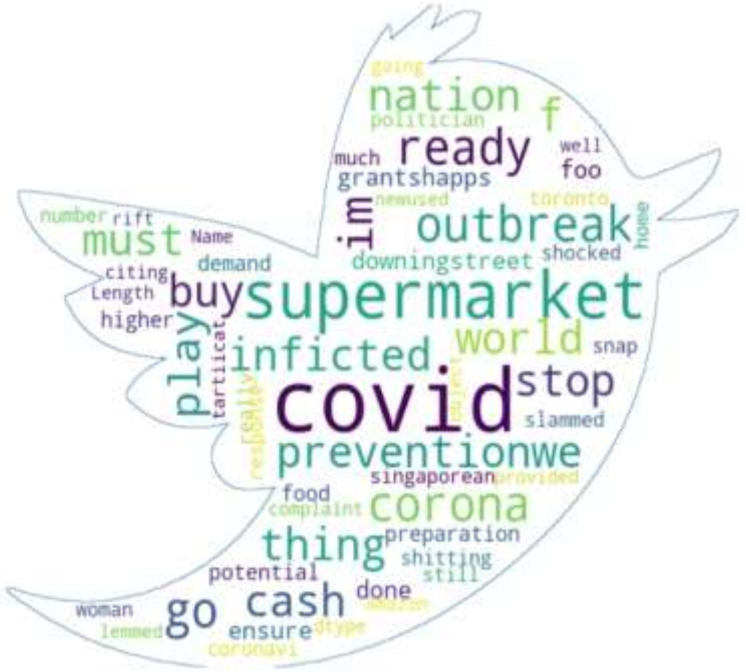
1. Positive Words



2. Neutral Words



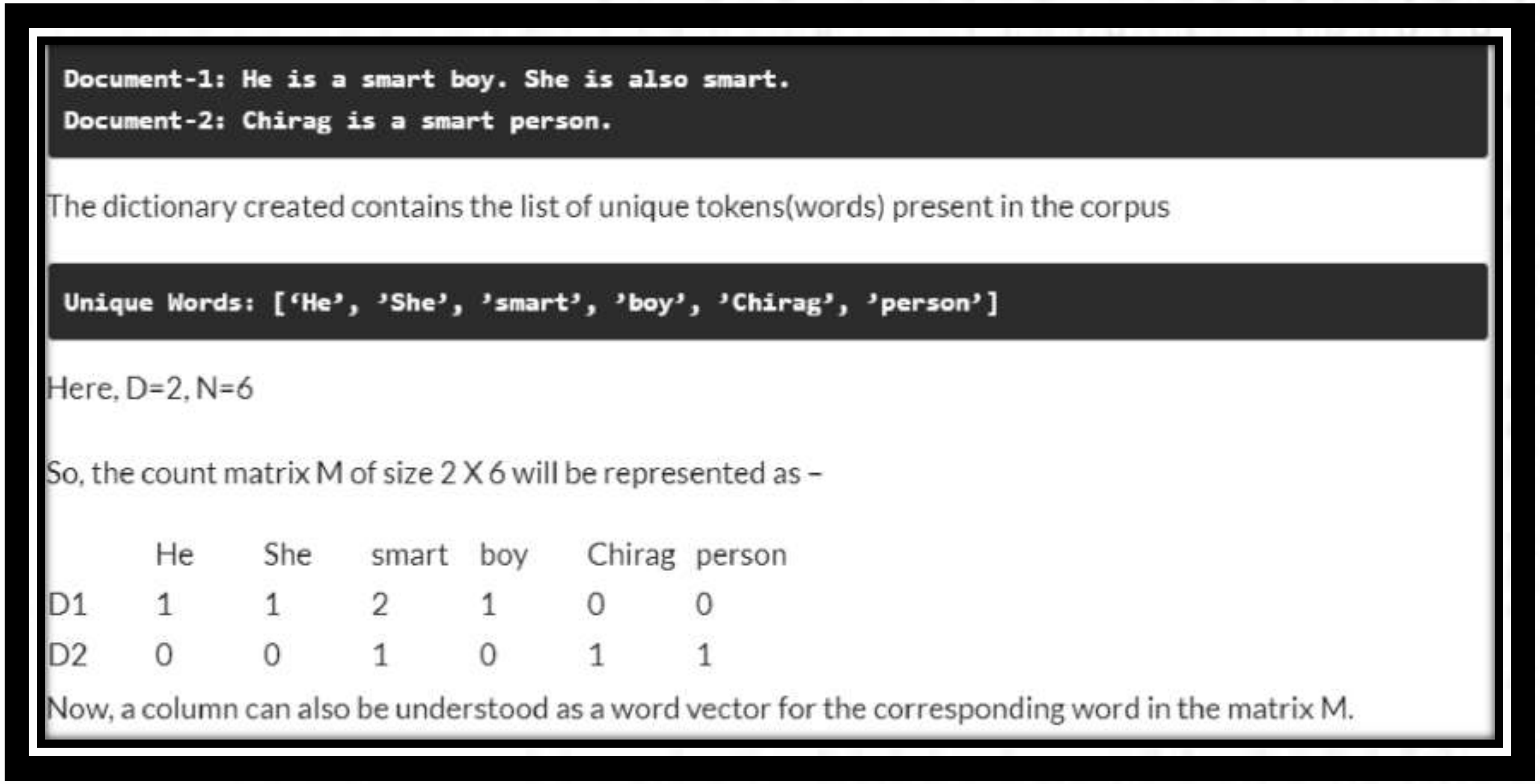
3.Negative Words



Model Preprocessing

Extracting Features From Text

- **Count Vectorizer -** CountVectorizer is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text.



```
Document-1: He is a smart boy. She is also smart.
Document-2: Chirag is a smart person.
```

The dictionary created contains the list of unique tokens(words) present in the corpus

```
Unique Words: ['He', 'She', 'smart', 'boy', 'Chirag', 'person']
```

Here, D=2, N=6

So, the count matrix M of size 2 X 6 will be represented as –

	He	She	smart	boy	Chirag	person
D1	1	1	2	1	0	0
D2	0	0	1	0	1	1

Now, a column can also be understood as a word vector for the corresponding word in the matrix M.

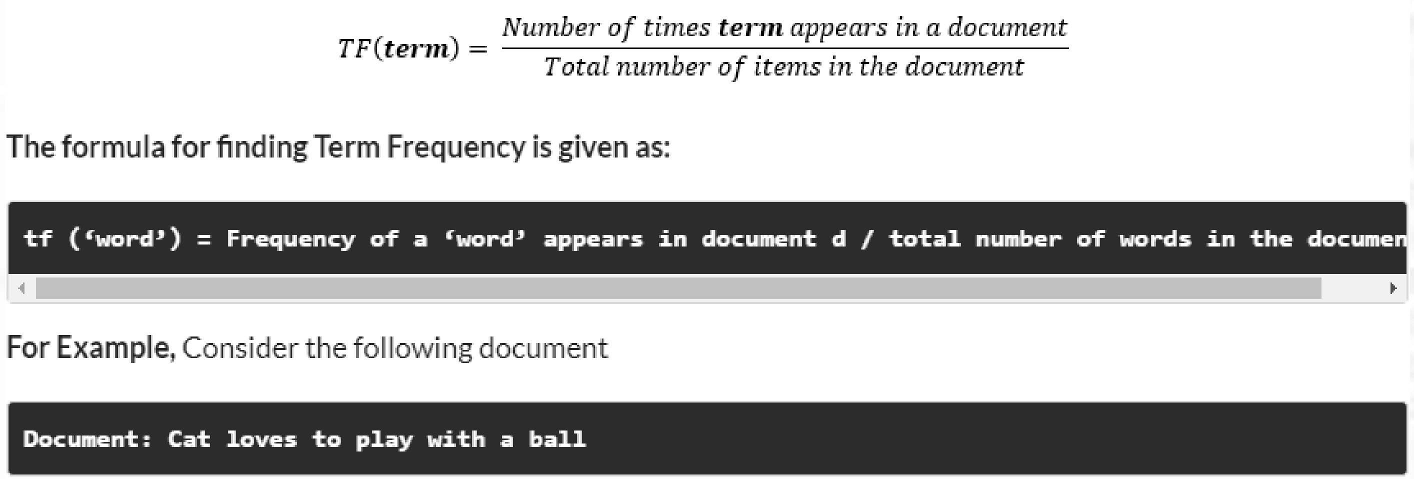
Model Preprocessing

Extracting Features From Text

- **TF-IDF -** Count Vectorizer method is simple and works well, but the problem with that is that it treats all words equally. As a result, it cannot distinguish very common words or rare words. So, to solve this problem, TF-IDF comes into the picture! Term frequency-inverse document frequency (TF-IDF) gives a measure that takes the importance of a word into consideration depending on how frequently it occurs in a document and a corpus.

TF – Term Frequency

Term frequency denotes the frequency of a word in a document.
It is the percentage of the number of times a word (x) occurs in a particular document (y) divided by the total number of words in that document



$$TF(\mathbf{term}) = \frac{\textit{Number of times } \mathbf{term} \textit{ appears in a document}}{\textit{Total number of items in the document}}$$

The formula for finding Term Frequency is given as:

```
tf ('word') = Frequency of a 'word' appears in document d / total number of words in the documen
```

For Example, Consider the following document

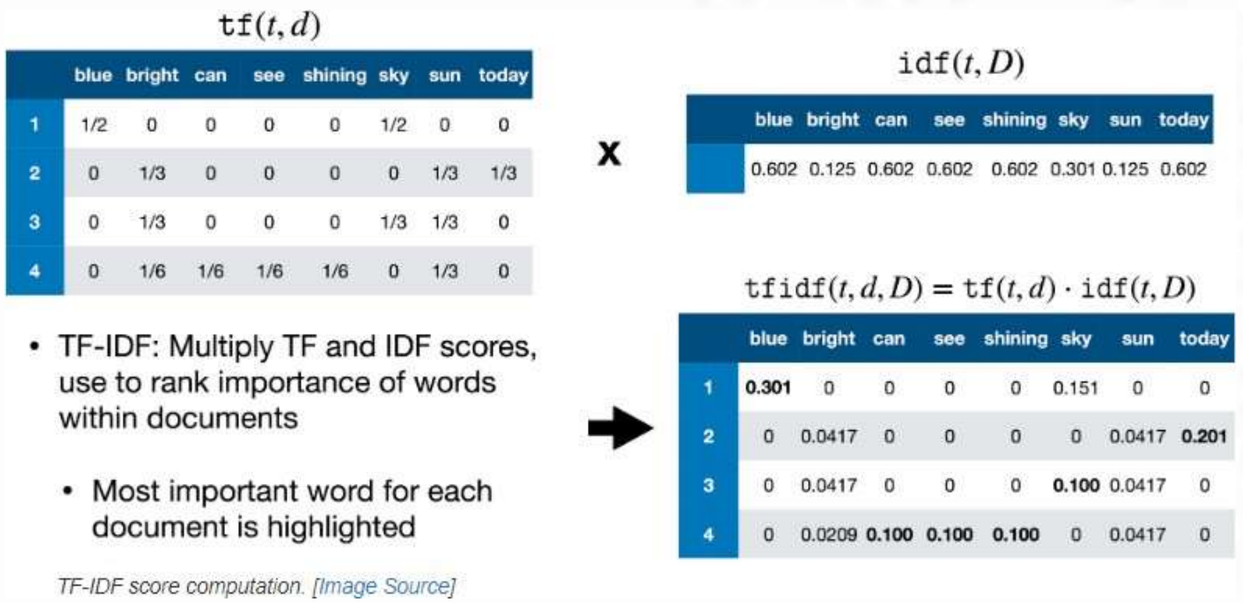
```
Document: Cat loves to play with a ball
```

Model Preprocessing

Extracting Features From Text

Inverse Document Frequency - It measures the importance of the word in the corpus. It measures how common a particular word is across all the documents in the corpus.
It is the logarithmic ratio of no. of total documents to no. of a document with a particular word.

The difference in the TF-IDF method is that each cell doesn't indicate the term frequency, but contains a weight value that signifies how important a word is for an individual text message or document



$\mathtt{tf}(t,d)$

	blue	bright	can	see	shining	sky	sun	today
1	1/2	0	0	0	0	1/2	0	0
2	0	1/3	0	0	0	0	1/3	1/3
3	0	1/3	0	0	0	1/3	1/3	0
4	0	1/6	1/6	1/6	1/6	0	1/3	0

x

$\mathtt{idf}(t,D)$

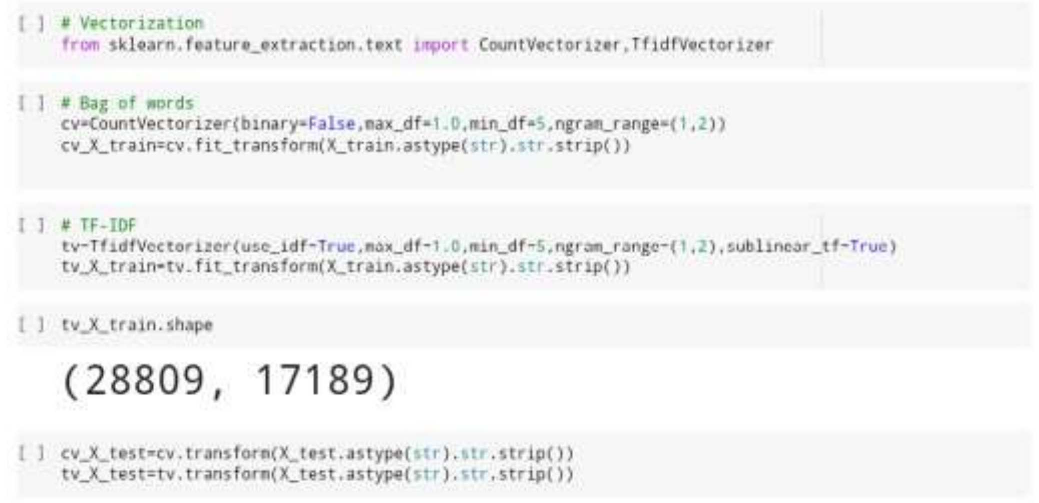
	blue	bright	can	see	shining	sky	sun	today
	0.602	0.125	0.602	0.602	0.602	0.301	0.125	0.602

$\mathtt{tfidf}(t,d,D) = \mathtt{tf}(t,d) \cdot \mathtt{idf}(t,D)$

	blue	bright	can	see	shining	sky	sun	today
1	**0.301**	0	0	0	0	0.151	0	0
2	0	0.0417	0	0	0	0	0.0417	**0.201**
3	0	0.0417	0	0	0	**0.100**	0.0417	0
4	0	0.0209	**0.100**	**0.100**	**0.100**	0	0.0417	0

- TF-IDF: Multiply TF and IDF scores, use to rank importance of words within documents
  - Most important word for each document is highlighted

TF-IDF score computation. [Image Source]



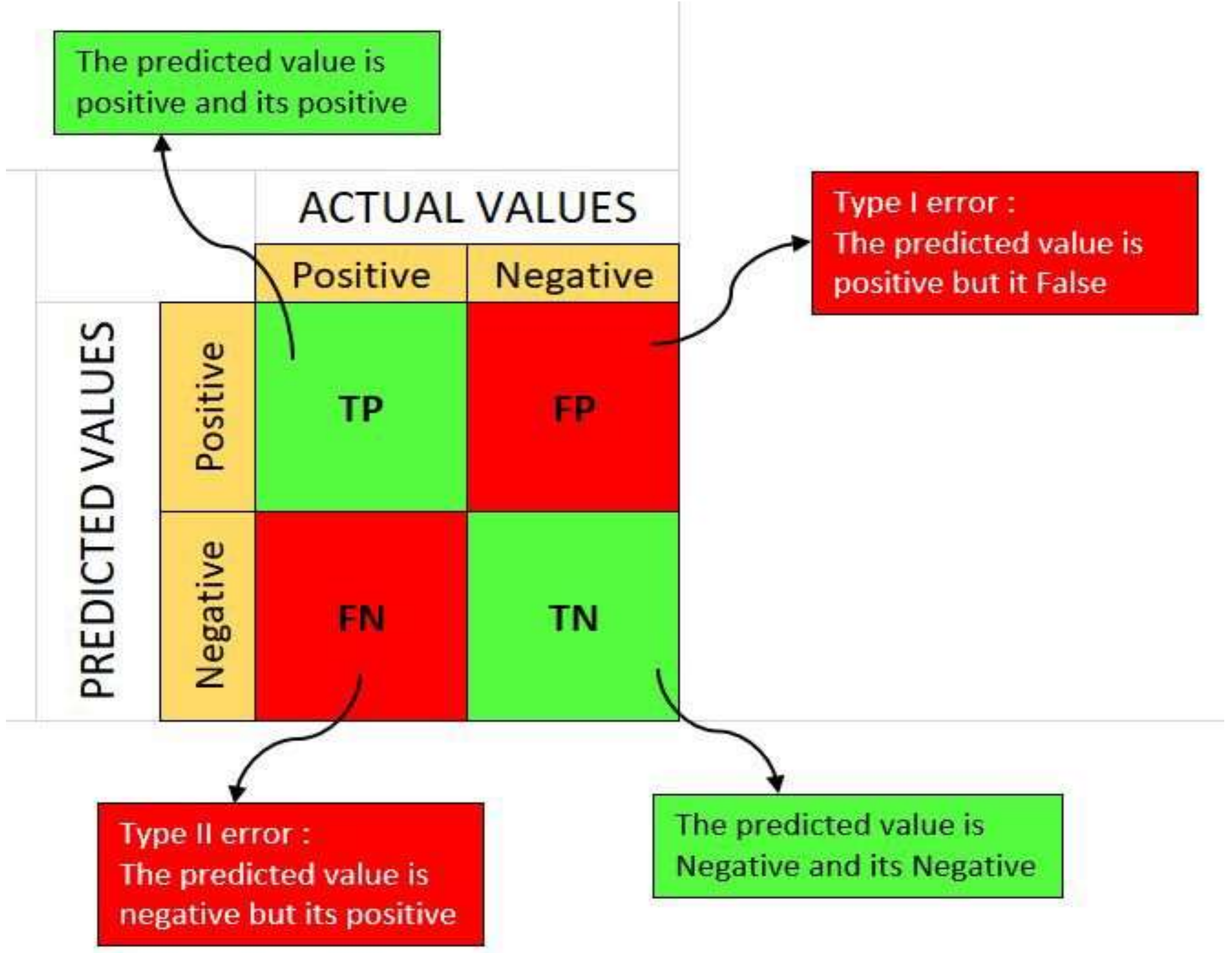
```
[ ] # Vectorization
    from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer

[ ] # Bag of words
    cv=CountVectorizer(binary=False,max_df=1.0,min_df=5,ngram_range=(1,2))
    cv_X_train=cv.fit_transform(X_train.astype(str).str.strip())

[ ] # TF-IDF
    tv=TfidfVectorizer(use_idf=True,max_df=1.0,min_df=5,ngram_range=(1,2),sublinear_tf=True)
    tv_X_train=tv.fit_transform(X_train.astype(str).str.strip())

[ ] tv_X_train.shape

    (28809, 17189)

[ ] cv_X_test=cv.transform(X_test.astype(str).str.strip())
    tv_X_test=tv.transform(X_test.astype(str).str.strip())
```

Model Training

Model Used

- Logistic Regression with Grid Search CV
- Decision Tree Classifier
- XG Boost Classifier
- KNN Classifier
- SVM Classifier

Performance metrics of classification models


The predicted value is positive and its positive
ACTUAL VALUES
Positive
Negative
PREDICTED VALUES
Positive
Negative
TP
FP
FN
TN
Type I error : The predicted value is positive but it False
Type II error : The predicted value is negative but its positive
The predicted value is Negative and its Negative

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

$$specificity = \frac{TN}{TN + FP}$$

Precision Accuracy And Recall

Precision

Precision is the proportion of correct predictions among all predictions of a certain class. In other words, it is the proportion of true positives among all positive predictions.

$$Precision = \frac{TP}{FP + TP}$$

Accuracy

Accuracy is the proportion of examples that were correctly classified. More precisely, it is sum of the number of true positives and true negatives, divided by the number of examples in the dataset.

$$Accuracy = \frac{TP + TN}{FP + FN + TP + TN}$$

Precision Accuracy And Recall

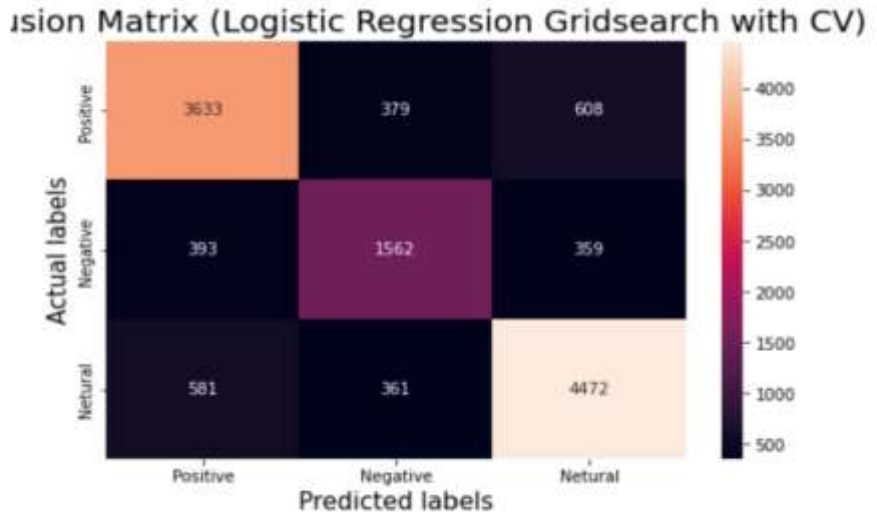
Recall

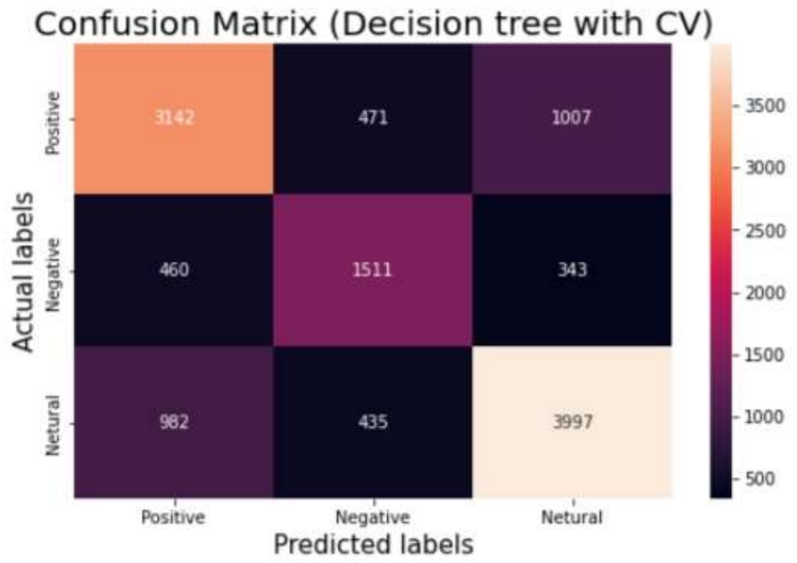
Recall is the proportion of examples of a certain class that have been predicted by the model as belonging to that class. In other words, it is the proportion of true positives among all true examples.

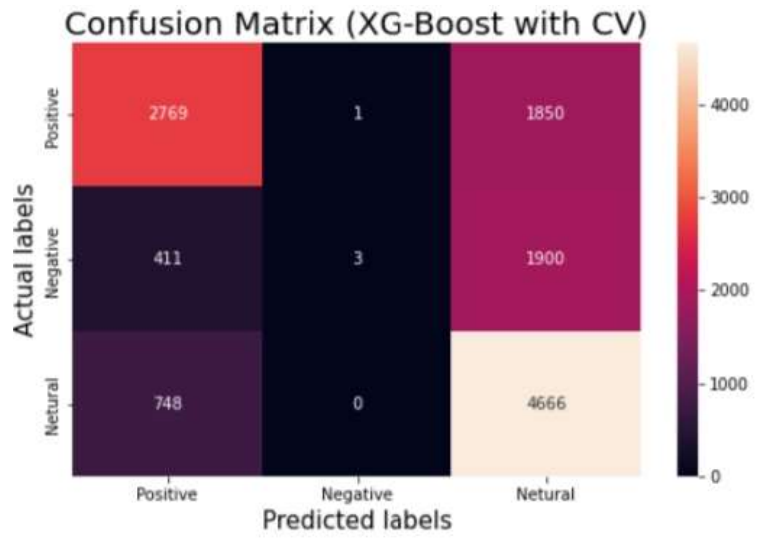
$$Recall = \frac{TP}{FN + TP}$$

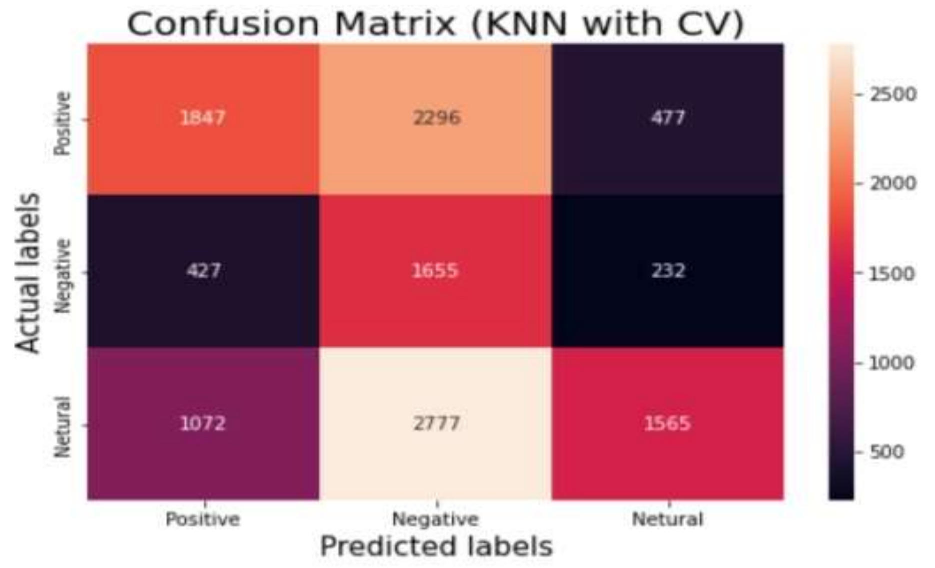
Confusion Matrix (count vector)

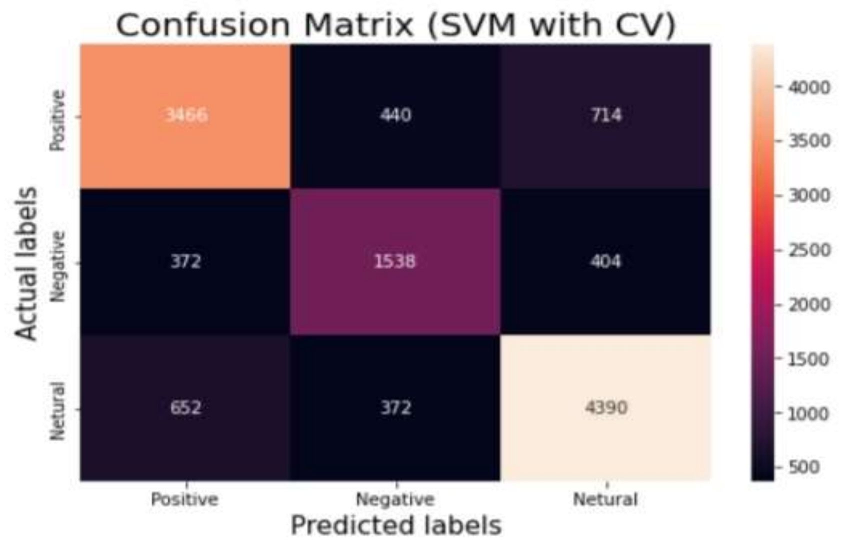
A **Confusion matrix** is an N x N matrix **used for** evaluating the performance of a classification model, where N is the number of target classes.



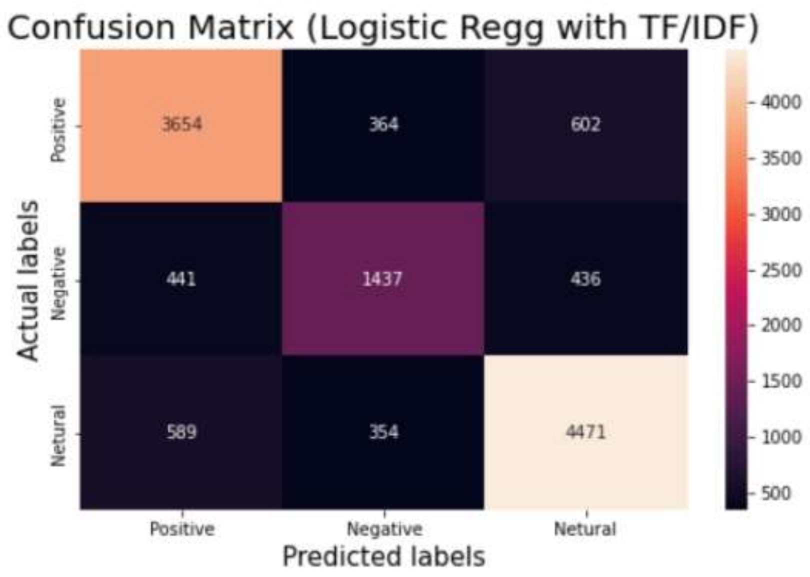


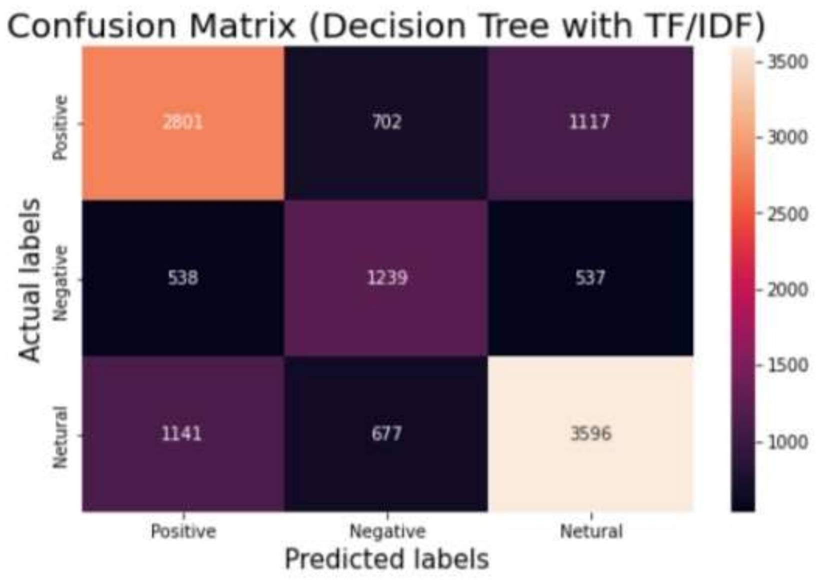


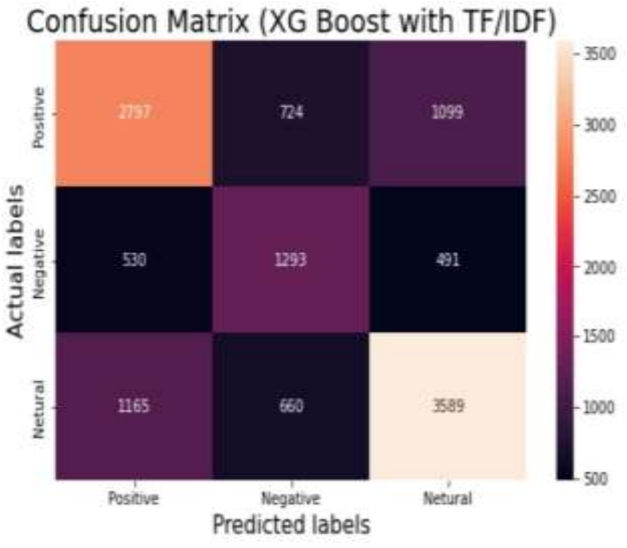


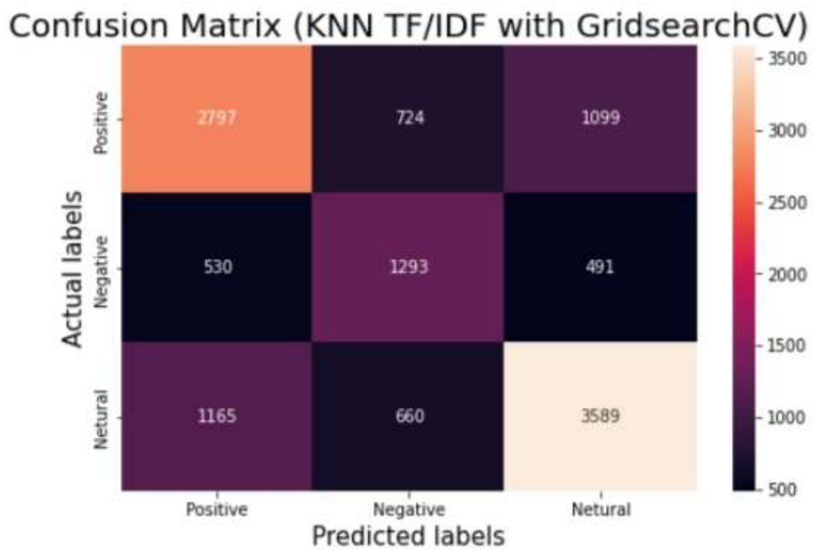


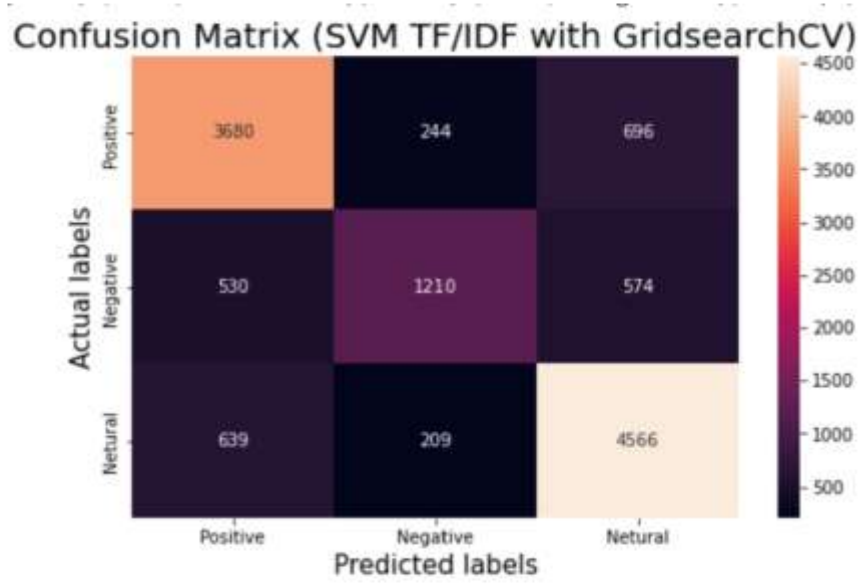
Confusion Matrix (TF-IDF Vector)











Performance Metrics and Accuracy (count vector)

Performance of Logistic Regression Model

	Precision	Recall	F1-score
Negative	0.79	0.79	0.79
Neutral	0.68	0.68	0.68
Positive	0.82	0.83	0.82
Accuracy			0.78

Performance of KNN Classifier

	Precision	Recall	F1-score
Negative	0.55	0.40	0.46
Neutral	0.25	0.72	0.37
Positive	0.69	0.29	0.41
Accuracy			0.41

Performance of Decision Tree Classifier

	Precision	Recall	F1-score
Negative	0.69	0.68	0.68
Neutral	0.62	0.67	0.64
Positive	0.75	0.74	0.75
Accuracy			0.70

Performance of SVM Classifier

	Precision	Recall	F1-score
Negative	0.77	0.75	0.76
Neutral	0.65	0.66	0.66
Positive	0.80	0.81	0.80
Accuracy			0.76

Performance of XG Boost Classifier

	Precision	Recall	F1-score
Negative	0.70	0.60	0.79
Neutral	0.75	0.00	0.68
Positive	0.55	0.86	0.67
Accuracy			0.60

Performance Metrics and Accuracy (TF/IDF Vector)

Performance of Logistic Regression Model

	Precision	Recall	F1-score
Negative	0.78	0.79	0.79
Neutral	0.66	0.62	0.64
Positive	0.81	0.82	0.82
Accuracy			0.77

Performance of KNN Classifier

	Precision	Recall	F1-score
Negative	0.37	1.00	0.55
Neutral	0.93	0.01	0.01
Positive	0.33	0.00	0.00
Accuracy			0.38

Performance of Decision Tree Classifier

	Precision	Recall	F1-score
Negative	0.62	0.60	0.61
Neutral	0.48	0.55	0.51
Positive	0.68	0.66	0.67
Accuracy			0.62

Performance of SVM Classifier

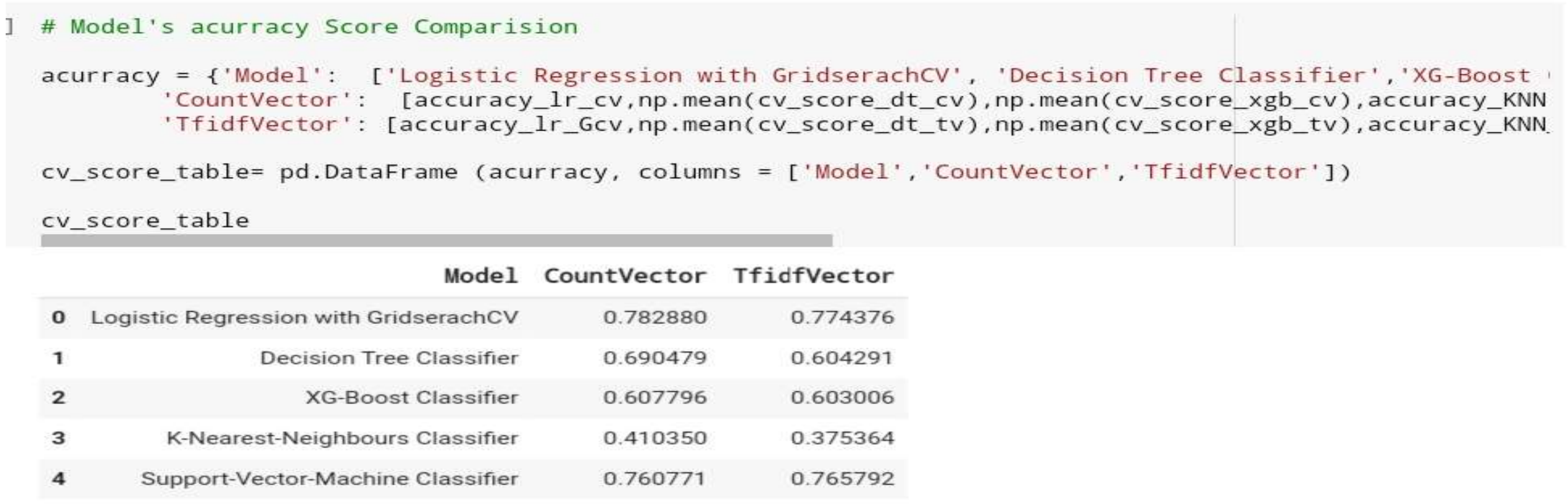
	Precision	Recall	F1-score
Negative	0.76	0.80	0.78
Neutral	0.73	0.52	0.61
Positive	0.78	0.82	0.81
Accuracy			0.77

Performance of XG Boost Classifier

	Precision	Recall	F1-score
Negative	0.63	0.61	0.62
Neutral	0.48	0.55	0.51
Positive	0.68	0.66	0.67
Accuracy			0.62

Conclusion

- We applied 5 models namely, Logistic Regression with Grid Search CV, Decision Tree Classifier, XG Boost, KNN, and SVM Classifier for both Count Vector And TF ID Vectorization techniques.
- We conclude that the machine is generating the best results for the Logistic Regression with Grid Search CV (count vectorizer) model with an Accuracy of 78.28% followed by the Logistic Regression with Grid Search CV (TF/ID vectorizer) model with an Accuracy of 77.43%.
- Also, we observed that no overfitting is seen for the data, and we can deploy this model.
- The sentiments of future tweets can be easily predicted using this model.
- Even being in the unprecedented situation of CoVid-19, people's positive sentiments outnumbered negative sentiments. However, negative sentiments also has a significant chunk which various Government agencies, NGOs, etc. can use to help boost the morale of the people and then in future repeat the analysis and comparing it with the present sentimental analysis to gauge the impact of the initiatives on the ground.



```
# Model's acurracy Score Comparision

acurracy = {'Model':  ['Logistic Regression with GridserachCV', 'Decision Tree Classifier','XG-Boost
        'CountVector':  [accuracy_lr_cv,np.mean(cv_score_dt_cv),np.mean(cv_score_xgb_cv),accuracy_KNN
        'TfidfVector': [accuracy_lr_Gcv,np.mean(cv_score_dt_tv),np.mean(cv_score_xgb_tv),accuracy_KNN_

cv_score_table= pd.DataFrame (acurracy, columns = ['Model','CountVector','TfidfVector'])

cv_score_table
```

	Model	CountVector	TfidfVector
0	Logistic Regression with GridserachCV	0.782880	0.774376
1	Decision Tree Classifier	0.690479	0.604291
2	XG-Boost Classifier	0.607796	0.603006
3	K-Nearest-Neighbours Classifier	0.410350	0.375364
4	Support-Vector-Machine Classifier	0.760771	0.765792

CHALLENGES FACED

- Text preprocessing.
- Vectorization.
- Model Training and performance improvement.

Thank you